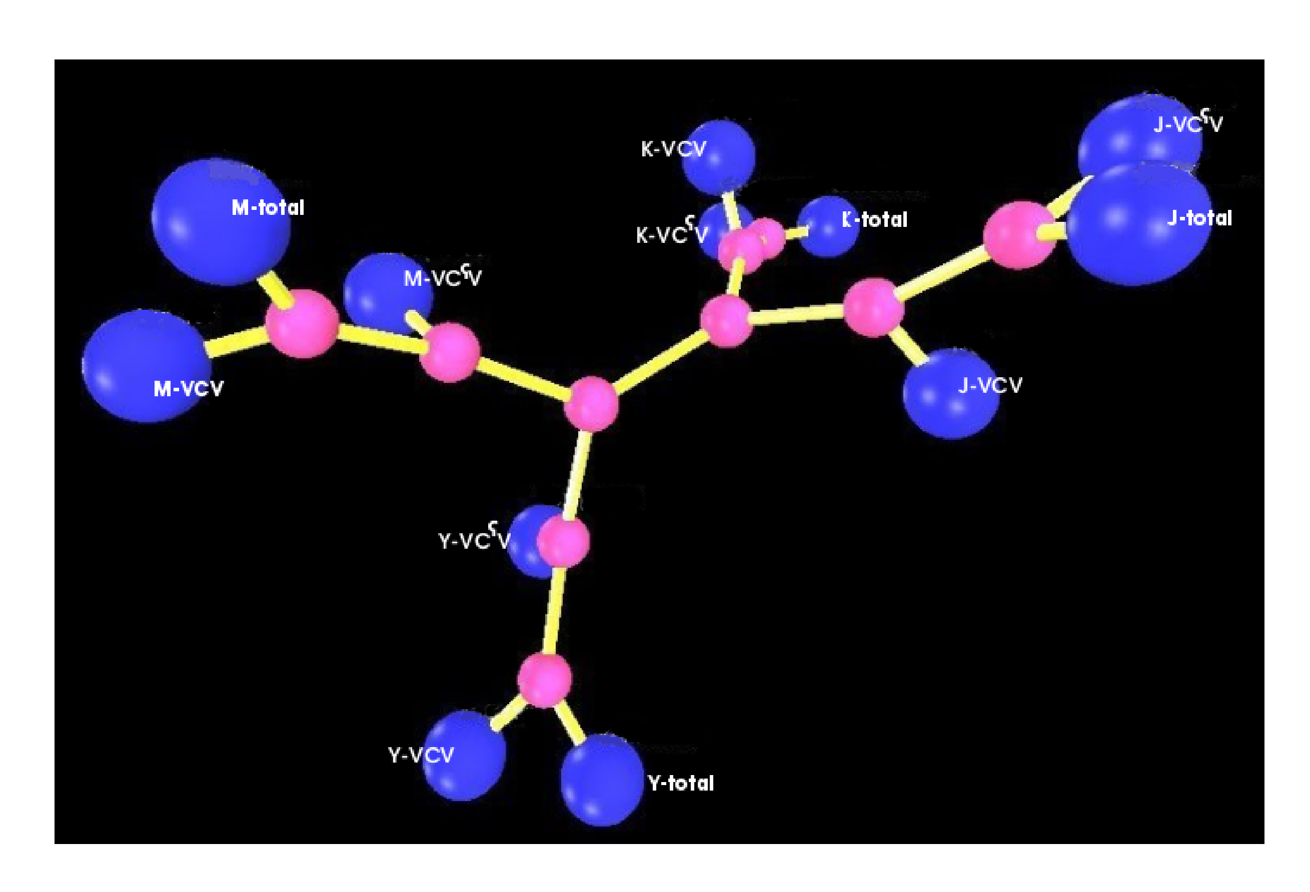




Franck LAURENT
Florian GOUDEY

Rapport de Projet : Analyse Temps/Fréquence



Licence Professionnelle SIL Option CDOOAM
Encadrant : Christian GUILLEMINOT

Promotion 2010





Sommaire





1 Références

Logiciels

- MPTK : https://gforge.inria.fr/projects/mptk/
- Cmake : http://www.cmake.org/
- Perl : http://www.perl.org/ (en) http://perl.enstimac.fr/ (fr)
- CPAN : http://www.cpan.org/ http://search.cpan.org/~grantm/XML-Simple-2.18/
- POD : http://perldoc.perl.org/perlpod.html
- PRAAT: http://www.fon.hum.uva.nl/praat/
- CompLearn : http://www.complearn.org/

Bibliographie

[1] R. Cilibrasi., The CompLearn Toolkit, http://complearn.sourceforge.net, 2003.

[2] R. Cilibrasi et P. Vitányi, Clustering by compression, IEEE Transactions on Information Theory, vol 51, pages 1523-1545, 2005.

[3] J.-P. Delahaye, Classer musiques, langues, images, textes et génomes, Pour la Science, 317:90-95, 2004.

[4] M. Embarki, M. Yeou, C. Guilleminot, S. Maqtari, An acoustic study of coarticulation in Modern Standard Arabic and Dialectal Arabic : pharyngealized vs non pharyngealized articulation, J. Rosenhouse (cord.), Special session : Arabic Phonetics at the Beginning of the 2nd Millenium, ICPhS, 6-10 August, Saarbrücken, Germany, 2007.

[5] A. Grossmann, J. Morlet Decomposition of Hardy functions into square integrable wavelets of constant shape, SIAM journal on mathematical analysis, vol. 15, n° 4, pages 723-736, 1984.

[6] A. Grossmann, B.Torrésani, http://www.cmi.univ-mrs.fr/~torresan/universalis/ondel.html

[7] C. Guilleminot, Décomposition adaptative du signal de parole appliquée au cas de l'arabe standard et dialectal, thèse de doctorat, Université de Franche-Comté, 2008.

[8] W. Heisenberg, Les principes physiques de la théorie des quanta, Gauthier-Villars,1932, réédition par Jacques Gabay, 1989.

[9] A. N. Kolmogorov, Three approaches for dening the concept of information quantity, Information Transmission, vol. 1, pages 311, 1965.

[10] S. Krstulovic' et R. Gribonval, The Matching Pursuit Tool Kit. INRIA, 2007.

[11] S. Krstulovic' et R.Gribonval, MPTK: Matching pursuit made tractable, Proc. Int. Conf. on Acoustic Speech and Signal Processing, 2006.

[12] S. Mallat et Z. Zhang : Matching pursuit with time-frequency dictionaries. IEEE Transactions on Signal Processing, 41:33973415, 1993.

[13] A. Moles, Information theory and esthetic perception. Urbana, University of illinois press, 1968.

[14] V. Popovici et JP. Thiran, Adaptive kernel matching pursuit for pattern classification, 14th International Conference on Artificial Intelligence and Applications (AIA), Innsbruck, Austria, 2004.

[15] M. Rocha Iturbide, Les techniques granulaires dans la synthèse sonore, thèse de doctorat, PARIS VIII, http://www.artesonoro.net/tesisgran/indicegran.html, 1999

[16] N. Wiener, Spatio-temporal continuity, quantum theory and music, 1964, In M. Capek, ed.,. The Concepts of space and Time, Boston: Reidel, 1975.

Mots clés

MPTK : Matching Pursuit ToolKit.

cross platform make.

Perl : langage de programmation.

CPAN : Comprehensive Perl Archive Network.

POD : Plain Old Documentation format.

PRAAT : logiciel scientifique (phonétique).

CompLearn.

XSLT : Extensible Stylesheet Language Transformations.

XML : Extensible Markup Language.

CSV : Comma-separated values.

7z.

LZMA : Lempel-Ziv-Markov chain-Algorithm.

Atome de Gabor.

2 Introduction

2.1 Présentation du contexte

Notre travail s'intègre dans une étude scientifique de la parole. Nous avons réalisé des outils permettant d'accélérer les expériences et les mesures liées à cette étude. Nous allons commencer par vous présenter cette étude.

2.1.1 Problématique

M. GUILLEMINOT nous propose, dans cette étude, d'adapter la décomposition granulaire du signal sonore par Matching Pursuit [12] à l'analyse phonétique. La décomposition granulaire est appliquée à des séquences V_1 C_2 V_2 ou plus longues, assortie d'une mesure de distance entre ces séquences ou des groupes de séquences basée sur la complexité de Kolmogorov [9]. Cette dernière permet un traitement des données par clustering, très simple et performant. M. GUILLEMNOT se base sur son étude originale[7].

Il explique que la méthode actuellement la plus utilisée en phonétique repose sur les séries de Fourier. Ces séries sont composées de fonctions sinusoïdes pour décrire les signaux à analyser. Or, l'onde sinusoïdale est par définition de durée infinie, d'intensité et fréquence constante quand les ondes réelles sont de durée limitées. M. GUILLEMINOT précise que les signaux réels, dont ceux de la parole sont continûment variables et exigent une adaptation des

séries de Fourier : ce sont les transformées de Fourier. Néanmoins, elles ne permettent pas d'échapper au paradoxe connu sous le nom de "principe d'incertitude d'Heisenberg" [8], selon lequel une précision temporelle entraîne un manque de précision en fréquence, et une précision en fréquence devra se passer d'une précision temporelle [16].

2.1.2 La décomposition granulaire ou atomique des sons

Selon M. GUILLEMINOT, les théoriciens de la mécanique quantique, à l'instar de Wiener [16] et de Gabor, montrent que le son peut-être approché sous un aspect granulaire ou ondulatoire comme la lumière, tandis que Moles [13] démontre et mesure l'aspect quantique de notre perception (quantum d'information).

La théorie granulaire ou atomique des sons est utilisée avec succès en composition-synthèse musicale [15]. Elle est également utilisée en compression du signal et en reconnaissance des formes [14].

Un fichier sonore numérisé est très pauvre en information, il est au niveau d'entropie maximal permettant de reconstituer le signal original. Pour aller plus loin dans l'analyse il est nécessaire d'atteindre un niveau supérieur d'information sur le signal ce qui est fait par les transformées de Fourier ou par une représentation de type musicale.

M. GUILLEMINOT cite A. Grossman, qui explique : « Qu'est-ce qu'une ondelette ? En schématisant à l'extrême, nous dirons qu'une ondelette est l'idéalisation mathématique d'une note de musique. » [6]. Il continue en expliquant que contrairement à un fichier numérisé, une partition musicale contient un grand nombre d'informations pertinentes sur le signal, les transformées de Fourier se trouvant à un niveau intermédiaire. Il se trouve également que la mesure de la complexité de Kolmogorov permet de distinguer des partitions musicales avec d'excellents résultats [3].

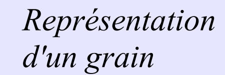
M. GUILLEMINOT propose donc de transformer les fichiers numérisés à analyser en sortes de partitions musicales mathématiques et de les distinguer par mesure de leur complexité.

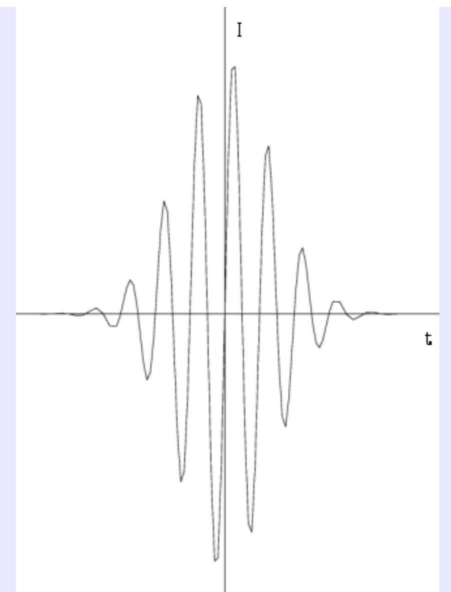
"L'analyse temps/fréquence"

Toujours selon M. GUILLEMINOT La décomposition granulaire repose sur l'utilisation de grains sonores (ondelettes, ou atomes) ayant une fréquence, une enveloppe (intensité, durée finie) que l'on adapte localement au signal. Les grains sont donc positionnés sur le plan temps/fréquence comme des notes de musique.









M. GUILLEMINOT indique que la première méthode de décomposition granulaire connue a été proposée par Morlet [5] et théorisée par Grossmann [6]. C'est une analyse multi-résolution à échelle logarithmique. Il continu en indiquant que Mallat [12] généralisera et assouplira la notion d'ondelettes avec l'algorithme adaptatif Macthing Pursuit (MP), méthode que M. GUILLEMINOT a choisie pour cette étude [10, 11].

"Le principe de Matching Pursuit"

Il explique que l'algorithme est récursif et qu'il est doté d'un dictionnaire contenant les formes des grains. Plus le dictionnaire est complet, plus la décomposition est précise. Soit un signal "S" nous fixons le nombre de boucles "nb" :

- le système choisit le grain "g" meilleur candidat avec le dictionnaire et l'adapte pour approcher le plus possible l'énergie maximale de "S";
- le grain est soustrait du signal et il est écrit dans un livre du signal. On a alors "R" = "S.g" où "R" est le reste du signal.
- "R" est alors renvoyé à l'étape 1 pour y subir le même traitement, et ainsi de suite jusqu'à "nb" boucles.

En fin d'analyse sont obtenus le livre des atomes du signal et un reste "R". La somme des atomes du livre donne le fichier original approché. La somme des atomes du livre + "R" donne exactement le fichier numérique original (resynthèse sans perte).

"Exploitation des fichiers d'atomes"

M. GUILLEMINOT précise que si "C" est la complexité de chaque livre relatif à une ou plusieurs séquences sonores, la mesure de cette complexité permet de connaître l'écart ou la distance entre ces séquences. M. GUILLEMINOT a utilisé la complexité de Kolmogorov [9]. Kolmogrov pose que si un fichier F1 est plus complexe qu'un fichier F2, le programme informatique pour générer F1 est plus long que celui permettant de générer F2. Il est facile de démontrer que la complexité de Kolmogorov est incalculable, mais M. GUILLEMINOT cite

que Cilibrasi & Vitaniy [2] ont montré que l'on pouvait la mettre en œuvre en compressant sans perte F1 et F2. Il continu en expliquant que le compresseur est alors considéré comme un langage ayant deux commandes ou instructions, « compresse » et « décompresse ». M. GUILLEMINOT explique que Delahaye [3] montre que l'erreur commise par cette méthode est très faible.

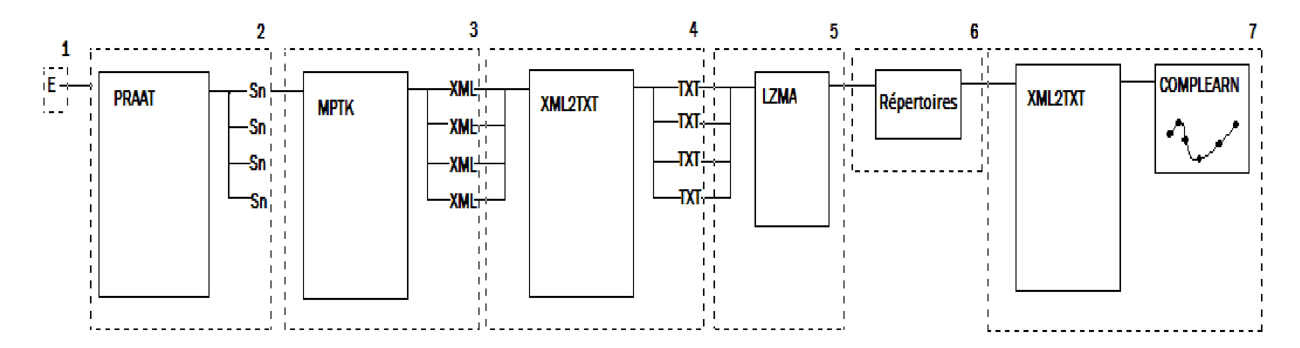
La première étude de M. GUILLEMINOT a utilisé le logiciel Complearn [1] dans sa version Windows(TM). Complearn calcule la distance entre les fichiers compressés deux à deux selon une équation appelée NCD (Normalized Compression Distance) et place les valeurs trouvées dans une matrice. Cette matrice permet de tracer le graphe des positions de chaque élément de signal traité.





2.2 Déroulement des expériences

L'étude présentée précédemment se décompose en plusieurs étapes :



1. nous partons d'un enregistrement sonore E contenant les séquences à comparer
2. les séquences sonores Sn sont extraites de E à l'aide du logiciel PRAAT et d'un script
3. les Sn sont décomposées avec MPTK et mises dans un format XML
4. les balises sont supprimées des fichiers XML pour qu'ils soient rendus en fichiers TXT
5. les fichiers TXT sont compressés par l'algorithme LZMA
6. ils sont assemblés et rangés selon le but à atteindre dans des répertoires
7. ils sont soumis par paquets à partir des répertoires à COMPLEARN pour classement, la sortie se fait sous formes de graphiques.

Pour améliorer le résultat, il faut obtenir la compression la plus efficace possible. D'où le choix d'utiliser LZMA ou 7Z.

2.3 Travail réalisé

Notre travail consiste à automatiser certaines de ces tâches. Nous avons commencé par automatiser l'étape 4 à l'aide d'un script PERL. Ensuite, nous avons automatisé les étapes 2 et 3 à l'aide d'un script propre au logiciel PRAAT. Ce script nous a permis d'intégrer notre premier travail. La méthodologie choisie nous permet d'anticiper l'étape 6. Cependant, par manque de temps, nous ne nous occuperons pas de l'étape 5.

A terme, le chainage de ces logiciels et le prototypage de la procédure décrite ci-dessus doivent aboutir à un outil scientifique utilisable.

Bien que tous les logiciels concernés soient multiplateformes, il nous a été demandé de valider le projet sous un environnement LINUX. Néanmoins, une adaptation multiplateformes reste envisageable. La principale difficulté est de faire fonctionner MPTK sous Windows™.

3 Travail préliminaire : installation de MPTK

L'une des consignes qui nous a été donnée était de créer un projet réutilisable par les différents utilisateurs de MPTK. Il nous a donc fallu compiler et installer ce logiciel pour





effectuer différents tests avec plusieurs dictionnaires pour mesurer la variété des fichiers XML que l'on peut rencontrer.

Il nous a fallu pour cela découvrir le moteur de production CMAKE et régler les disfonctionnements liés aux différentes versions de CMAKE.

3.1 Générer les informations de compilation avec CMAKE

La première étape sert à rassembler les informations nécessaires à la compilation. Un certain nombre de librairies doivent être installé au préalable. Celles-ci sont listées dans la documentation MPTK.

en mode console, sélectionner le répertoire de destination comme répertoire courant taper la commande « cmake [dossier_source] ». [dossier_source] est le dossier où les sources ont été désarchivées.

La version 2.6.4 de CMAKE renvoie une erreur qui n'existe pas avec la version 2.6.3 et qui n'est qu'un avertissement avec la version 2.6.8

CMake Error at src/CMakeLists.txt:7 (ADD_SUBDIRECTORY):The binary directory

/home/flaurent/projet/compil263/src/libmptk

is already used to build another source directory, so it cannot be used tobuild source directory

/home/flaurent/projet/MPTK-0.5.6/src/libmptk

Specify a unique binary directory name.

CMake Error at src/CMakeLists.txt:10 (ADD_SUBDIRECTORY):The binary directory

/home/flaurent/projet/compil263/src/libdsp_windows

is already used to build another source directory, so it cannot be used to build source directory

/home/flaurent/projet/MPTK-0.5.6/src/libdsp_windows

Specify a unique binary directory name.

Il faut mettre en commentaire les lignes 329 et 330 du fichier [dossier_source]/CmakeList.txt. On les met en commentaire en ajoutant le caractère « # » au début de la ligne.

3.2





3.3 Compilation des sources

Sur la même console taper « make »

Le logiciel est utilisable en l'état en le lançant dans le répertoire de compilation. Il suffit de taper « [dossier_source]/bin/mpd <option> ».

Il est possible de l'installer dans le système en tapant «sudo make install » et en tapant le mot de passe administrateur. Par défaut le logiciel s'installe dans « /usr/local ».Les librairies présente dans« [dossier_source]/lib/ » doivent être copiées manuellement dans « /usr/local/lib/ ». On lance alors le logiciel en tapant « mpd <option> »

Deux messages d'erreur sont encore susceptibles d'apparaître lors du lancement du logiciel

mpd: error while loading shared libraries: libmptk.so : cannot open shared object file: No such file or directory

Taper la commande « export LD_LIBRARY_PATH = "/usr/local/lib" » ou « export LD_LIBRARY_PATH = "[dossier_source]/lib » suivant l'emplacement depuis lequel vous lancez le logiciel.

mptk ERROR -- MPTK_Env_c::get_configuration_file() - Could not find MPTK Env variable with path to config file.
mptk ERROR -- MPTK_Env_c::load_environment() - couldn't load the MPTK environment

Taper la commande « export MPTK_CONFIG_FILENAME = "/usr/local/bin/path.xml" » ou « export MPTK_CONFIG_FILENAME = "/[dossier_source]/bin/path.xml" » suivant l'emplacement depuis lequel vous lancez le logiciel.

3.4 Utilisation de MPTK

Les livres d'atomes, par défaut, sont rendu de façon binaire dans un fichier indiqué par l'option "-D *fichier.bin*". Dans notre cas il faut noter "-D -". puis rediriger "> *fichier.xml*".

A ce stade du projet nous avions différents types de fichiers XML (voir annexe 2). Nous allions pouvoir nous pencher sur l'étape d'épuration de ces fichiers. Néanmoins, il est difficile d'en tirer un bilan précis : nous ne maîtrisons pas la diversité possible des dictionnaires.

4 Traitement des fichiers XML

4.1 Choix technologique

Notre expérience dans le traitement de fichiers, à ce moment du projet, était très succincte. Mr GUILLEMINOT nous a orientés vers un script PERL. La littérature disponible sur le web nous a encouragés sur cette voie.

De plus, la richesse des modules disponibles pour ce langage, nous a grandement facilité la tâche. Pour notre part, nous avons choisi le module XML::SIMPLE. Il hérite de XML::SAX qui a été la première solution à nous interpeler.

Le module XML::SIMPLE nécessite un contenu XML en entrée. Le traitement de ce contenu nous renvoie une variable sous forme de table de hachage contenant des listes ou des tables de hachage suivant s'il s'agit d'attribut de balise ou de contenu de balise (exemple en annexe 7).

4.2 Fonctionnalité du script xml2txt.pl

Le manuel d'utilisation est disponible en annexe 3.

Le fichier de sortie est un fichier texte avec séparateur. Les données de chaque atome sont écrites sur une ligne et séparées par une virgule.

Il est possible de ne récupérer que les données sous forme de contenu de balise ou de récupérer aussi celles sous forme d'attribut de balise.

Il est possible d'intégrer une ligne de titre et d'ajouter à chaque ligne une numérotation qui tient lieu d'identifiant unique. Dans le premier cas cité précédemment, les titres sont obtenus par l'attribut type de la balise s'il existe, sinon on reprend le nom de la balise. Dans le deuxième cas, les attributs portent leur propre nom ainsi que les balises.

Le script peut traiter un fichier XML donné ou tous les fichiers d'un dossier donné (par défaut le dossier courant).

Un sous dossier "resultat" est créé dans le même emplacement que le ou les fichiers traités. Les fichiers de sortie y sont rangés. Ils portent le même nom que les fichiers traités suivis de l'extension ".txt" (exemple : "test.XML" renvoi "test.XML.txt") comme le montre les trois images ci-dessous.





XML - Gestionnaire de fichiers

Fichier Éditer Vue Aller Aide

florian | projet | script_praat | test | XML

florian
Corbeille
Bureau
Système de fichiers
Lecteur de disquettes
Documents
Téléchargement
Musique
Images
Vidéos

1_aspa.xml 2_aspa.xml 3_aspa.xml
4_i__fdpi.xml 5_i__fdpi.xml 6_i__fd_9ei.xml
7_i__fs_9ei.xml 8_i__fs_9ei.xml 9_i__fs_9ei.xml

XML - Gestionnaire de fichiers

Fichier Éditer Vue Aller Aide

florian | projet | script_praat | test | XML | resultat

florian
Corbeille
Bureau
Système de fichiers
Lecteur de disquettes
Documents
Téléchargement
Musique
Images
Vidéos

resultat 1_aspa.xml 2_aspa.xml
3_aspa.xml 4_i__fdpi.xml 5_i__fdpi.xml
6_i__fd_9ei.xml 7_i__fs_9ei.xml 8_i__fs_9ei.xml

resultat - Gestionnaire de fichiers

Fichier Éditer Vue Aller Aide

florian | projet | script_praat | test | XML | resultat

florian
Corbeille
Bureau
Système de fichiers
Lecteur de disquettes
Documents
Téléchargement
Musique
Images
Vidéos

1_aspa.xml.txt 2_aspa.xml.txt 3_aspa.xml.txt
4_i__fdpi.xml.txt 5_i__fdpi.xml.txt 6_i__fd_9ei.xml.t
xt

4.3 Fonctionnement du script

Le script est disponible en annexe 4.

Pour commencer le script traite les arguments entrés. Il tolère que ceux-ci soit rentrés dans n'importe quel ordre. Ensuite, le script crée le dossier "resultat" et crée le fichier de sortie. Après cette phase, le script crée le parseur XML (XML::SIMPLE), ouvre le fichier de sortie et appel la fonction de traitement de table de hachage. Une fois le traitement terminé, le résultat est enregistré dans le fichier de sortie qui est ensuite fermé. Dans le cas du traitement d'un dossier complet, le script réinitialise certaine variable pour recommencer avec un nouveau fichier XML.

La variable rendu par le parseur est une table de hachage contenant des listes et des tables de hachage. En conséquent, le traitement de cette variable nécessite deux sous routines itératives. La première traite toutes les tables de hachage rencontrées et l'autre traite les listes. Ces deux sous routines s'appellent l'une est l'autre jusqu'à obtenir les informations pertinentes. Ces informations sont toujours dans une table de hachage. Une gestion du niveau d'itération permet de gérer les sauts de ligne et la numérotation si elle est demandée.

4.4 Bilan

Initialement, notre projet devait s'arrêter là. Au vue du temps qu'il nous a fallu, Mr GUILLEMINOT nous a proposés d'aller plus loin en automatisant d'autres étapes de son travail.

D'autre part, notre enseignement théorique s'est poursuivit en parallèle, notamment concernant le traitement de fichier XML. Si nous devions refaire l'étude nous mettrions en concurrence la solution d'utiliser XSLT avec une feuille de style XLS.

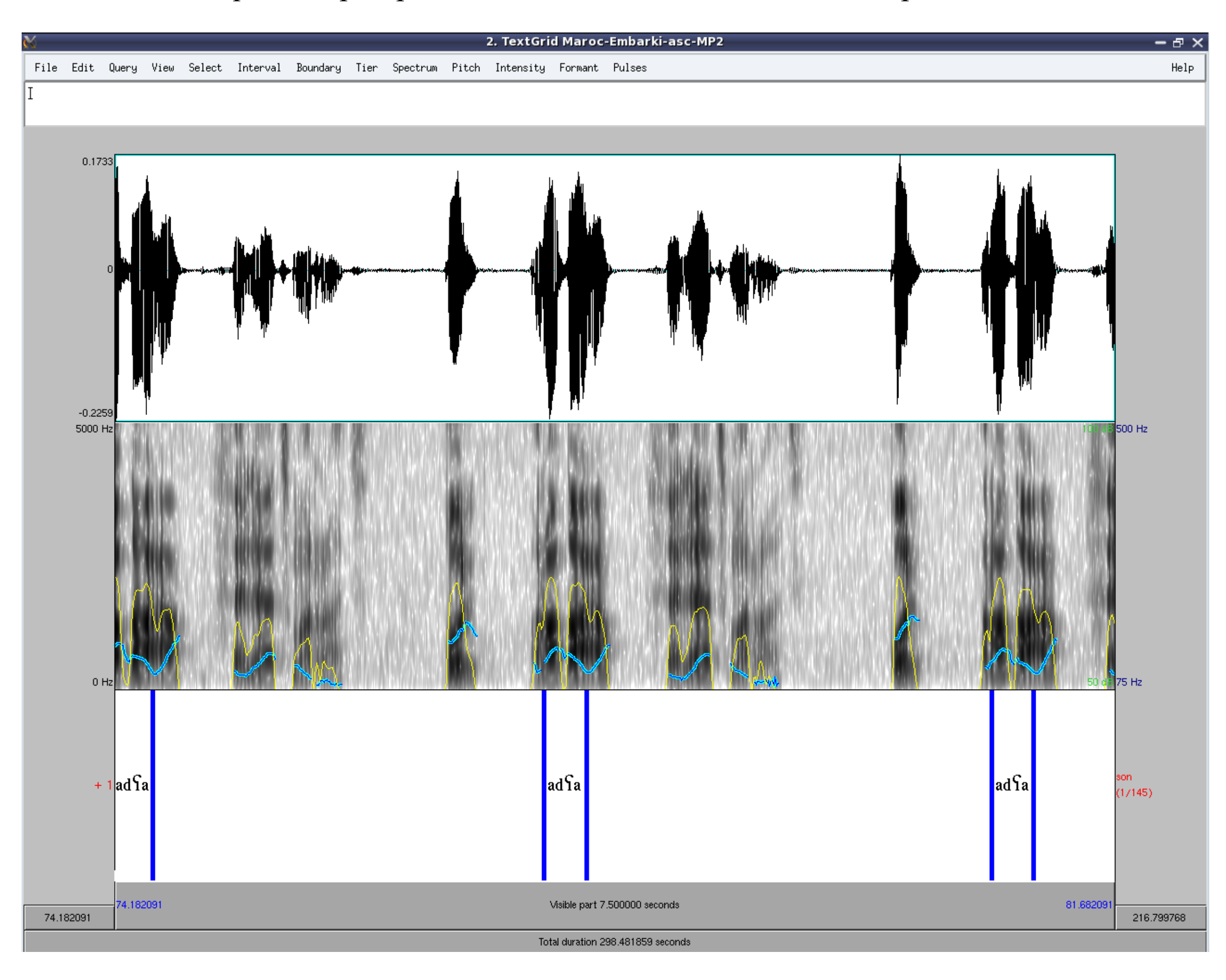




5 Découpe des fichiers WAV et traitement systématique par MPTK

5.1 Choix technologique

La solution qui au départ paraissait évidente était d'utiliser un script BASH.

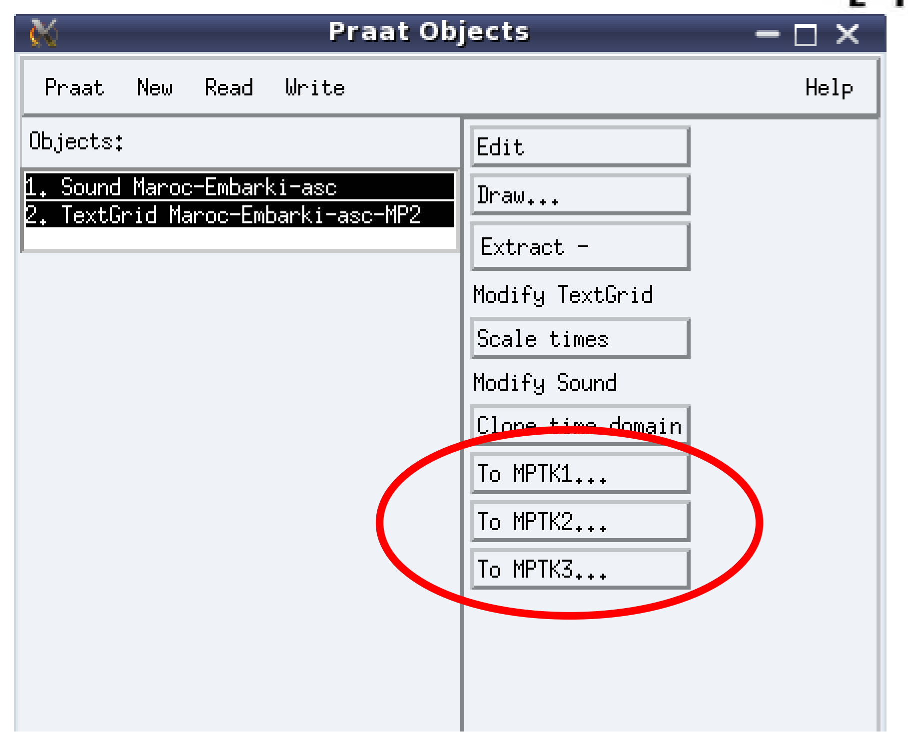


Les séquences sonores, au sein de l'enregistrement, sont identifiées grâce au logiciel PRAAT et mémorisées dans un TEXTGRID (simple fichier texte doté d'une extension ".TextGrid", voir annexe 8). Or, ce logiciel dispose de son propre script qui permet :

- d'utiliser toutes les fonctions propres à PRAAT
- d'envoyer des commandes au système SHELL
- de créer des fenêtres pour que l'utilisateur puisse rentrer certains paramètres
- des appels de script via des boutons créés sur l'interface graphique principale.







Ce script dispose d'autres fonctionnalités qui ne nous sont pas utiles. La facilité de mise en œuvre pour différents utilisateurs nous a conduits à valider ce choix.

5.2 Fonctionnement des scripts

Le manuel est disponible en annexe 5 et les scripts en annexe 6.

Trois variantes ont été réalisées pour permettre différents essais au sein de l'étude globale.

5.2.1 Script initial

Le script n'est fonctionnel que si l'utilisateur sélectionne un objet de type "Sound" (enregistrement E) et un objet de type "TextGrid" (condition pour que le bouton d'appel apparaisse). Au démarrage, l'utilisateur est invité à rentrer certaines informations liées aux commandes système utilisées. Le script commence par extraire toutes les séquences sonores définis dans la TextGrid. Ensuite, il prend les nouveaux objets créés un par un :

- il les enregistre dans des fichiers WAV
- il lance le traitement par MPTK
- il lance le traitement par xml2txt.pl
- il supprime l'objet de la liste PRAAT.

5.2.2 Variantes

La première variante consiste à resampler les objets Sound créés avant de les enregistrer en fichiers WAV.

Pour la seconde variante, tous les objets portant le même nom sont concaténés dans le même fichier WAV. Ceci est réalisé grâce à une fonction existante de PRAAT. Le traitement par MPTK ne peut plus être fait immédiatement, il faut donc attendre que tous les fichiers





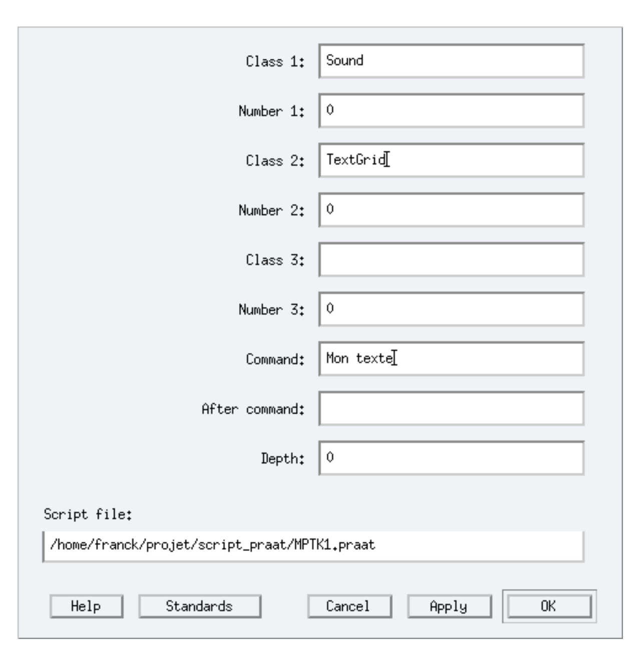
WAV soient créés pour générer une liste de fichiers WAV rangée dans notre dossier de destination grâce à une fonction propre à PRAAT. Ensuite il faut les traiter un à un.

5.3 Création un bouton dans le menu dynamique

1. Ouvrez le script que vous voulez lancer par un bouton : *menu Praat/Open Praat Script...*
2. La fenêtre d'édition de script s'ouvre.
3. Créez le bouton : *menu File/Add to dynamic menu...*
4. Une boite de dialogue apparaît.



5. Dans les cases *Class* remplissez les différents type d'objets à sélectionnés pour voir apparaître le bouton.







6. Dans la case *Command* taper le texte que vous voulez voir affiché sur le bouton
7. Cliquez sur *OK*, fermez la fenêtre d'édition de script et testez.

5.4 Bilan

Outre le fait que ces variantes serviront pour les études de Mr GUILLEMINOT, elles nous ont servies à traquer un défaut. Au final ce défaut s'est avéré être lié au dictionnaire utilisé pour MPTK.

Ce défaut générant des fichiers XML vides, il a été souhaitable d'utiliser le script xml2txt.pl en mode fichier par fichier et non en mode dossier pour faciliter l'analyse.

Cette solution est également gardée dans le but de débuguer d'éventuels dictionnaires défectueux.

Par manque de temps, notre projet se termine à la fin de cette partie.

6 Réalisation de la documentation

6.1 Choix de l'outil

L'utilisation de PERL nous a orientés vers l'outil POD. Il s'agit d'un langage de description qui permet de générer une documentation dans différents formats (HTML, MAN, TXT,…). Cet outil est installé automatiquement avec PERL.

6.2 Réalisation

Cet outil étant natif de PERL, le code POD est intégré directement dans le code PERL (voir annexe 4). Pour les scripts PRAAT, il nous a fallu créer un fichier "MPTK1.pl" indépendant des fichiers ".praat". Pour que cette documentation soit la plus compréhensible possible, nous avons repris les conventions usuelles des documentations PERL. En générale, elle se compose de :

- NOM
- SYNOPSIS
- DESCRIPTION
- AUTEUR

La première partie "NOM" donne le nom du script et un rapide descriptif de son rôle. La deuxième partie décrit la ligne de commande pour exécuter le script. Les arguments sont détaillés dans la partie "DESCRIPTION". La dernière partie présente le ou les auteurs du script.

6.3 Utilisation de POD

POD s'utilise en ligne de commande. Il existe plusieurs commandes suivant le format de sortie souhaité (par exemple : pod2html pour un fichier de sortie en HTML). Dans le cas de fichiers HTML, nous n'avons pas trouvé d'option permettant de noter l'encodage directement en "ISO-8859-1" (codage de l'Europe occidentale). Nous avons donc effectué cette opération manuellement avec un éditeur HTML.

7 Conclusion

Ce projet nous a apporté un certain nombre de connaissances, il nous a notamment familiarisés avec l'environnement LINUX et les installations complètes depuis le code source d'un logiciel. Il nous a aussi apporté une expérience dans l'utilisation de fichier XML et dans la recherche de documentation technique. En effet nous avons pu aborder différents logiciels comme MPTK ou PRAAT et différents langages comme PERL, POD ou le script PRAAT.

Nous avons pu mesurer aussi la notion de double compétence inhérente au métier d'informaticien. Nous avons du intégrer des connaissances propre à la phonétique ou à l'étude des ondes.

Nous remercions Monsieur Christian Guilleminot de nous avoir proposé ce projet, son soutien et ses précieux conseils dans celui-ci. Nous avons apprécié nos rapports avec lui, qui étaient des rapports très professionnels de coéquipier plutôt que d’étudiants à encadrant.





8 Annexes

Sommaire





Annexe 1 : Exemples de livre d'atomes

Exemple de livre d'atomes n°1


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book nAtom="4" numChans="1" numSamples="262144" sampleRate="44100"
libVersion="0.5.6">
	<atom type="gabor">
		<par type="numChans">1</par>
		<support chan="0"><p>139264</p><l>8192</l></support>
		<par type="amp" chan="0">22.8102</par>
		<window type="gauss" opt="0"></window>
		<par type="freq">0.0854492</par>
		<par type="chirp">0</par>
		<par type="phase" chan="0">1.10209</par>
	</atom>
	<atom type="gabor">
		<par type="numChans">1</par>
		<support chan="0"><p>172032</p><l>8192</l></support>
		<par type="amp" chan="0">15.7853</par>
		<window type="gauss" opt="0"></window>
		<par type="freq">0.0854492</par>
		<par type="chirp">0</par>
		<par type="phase" chan="0">-0.433481</par>
	</atom>
	<atom type="gabor">
		<par type="numChans">1</par>
		<support chan="0"><p>90112</p><l>8192</l></support>
		<par type="amp" chan="0">14.3264</par>
		<window type="gauss" opt="0"></window>
		<par type="freq">0.0477295</par>
		<par type="chirp">0</par>
		<par type="phase" chan="0">-1.91202</par>
	</atom>
	<atom type="gabor">
		<par type="numChans">1</par>
		<support chan="0"><p>94208</p><l>8192</l></support>
		<par type="amp" chan="0">13.0344</par>
		<window type="gauss" opt="0"></window>
		<par type="freq">0.0477295</par>
		<par type="chirp">0</par>
		<par type="phase" chan="0">2.52425</par>
	</atom>
</book>
```





Exemple de livre d'atomes n°2


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book nAtom="3" numChans="1" numSamples="262144" sampleRate="44100"
libVersion="0.5.6">
        <atom type="anywavehilbert">
                <par type="numChans">1</par>
                <support chan="0"><p>140589</p><l>64</l></support>
                <par type="amp" chan="0">2.39254</par>
                <par
type="filename">/home/franck/projet/compil/bin/reference/wavetable/defaultWaveTa
ble.xml</par>
                <par type="filterIdx">8</par>
                <anywavePar chan="0">
                        <par type="realPart">-0.183622</par>
                        <par type="hilbertPart">-0.982997</par>
                </anywavePar>
        </atom>
        <atom type="anywavehilbert">
                <par type="numChans">1</par>
                <support chan="0"><p>140665</p><l>64</l></support>
                <par type="amp" chan="0">2.1013</par>
                <par
type="filename">/home/franck/projet/compil/bin/reference/wavetable/defaultWaveTa
ble.xml</par>
                <par type="filterIdx">8</par>
                <anywavePar chan="0">
                        <par type="realPart">0.152159</par>
                        <par type="hilbertPart">0.988356</par>
                </anywavePar>
        </atom>
        <atom type="anywavehilbert">
                <par type="numChans">1</par>
                <support chan="0"><p>140532</p><l>64</l></support>
                <par type="amp" chan="0">2.04635</par>
                <par
type="filename">/home/franck/projet/compil/bin/reference/wavetable/defaultWaveTa
ble.xml</par>
                <par type="filterIdx">8</par>
                <anywavePar chan="0">
                        <par type="realPart">-0.635876</par>
                        <par type="hilbertPart">-0.771791</par>
                </anywavePar>
        </atom>
</book>
```





Annexe 2 : Manuel de xml2txt.pl

NOM

xml2txt.pl - Transforme les livres d'atomes XML de MPTK en fichiers texte exploitables par d'autres logiciels.

SYNOPSIS

xml2txt.pl [t] [c] [a] [fichierXML] [repertoire]

DESCRIPTION

Sans argument cette commande traite l'intégralité des fichiers XML du répertoire courant. Les fichiers résultats sont générés dans un répertoire resultat à l'adresse du ou des fichiers XML traités. Ces fichiers portent le même nom que le fichier XML plus l'extension txt.

Les options suivantes sont possibles :

t intègre une ligne de titre pour un traitement par tableur par exemple.

c intègre une colonne de numérotation pour un traitement par tableur par exemple.

a permet de traiter les attributs de balise XML en plus du contenu des balises.

fichierXML : il permet de ne traiter qu'un seul fichier. Il peut être indiqué en adresse relative ou absolue. La génération du résultat répond aux mêmes règles que dites précédemment. Cette option ne peut être utilisée qu'une seule fois.

repertoire : permet de traiter un répertoire autre que le répertoire courant. La génération des résultats répond aux mêmes règles que dites précédemment. Cette option ne peut être utilisée qu'une seule fois et sans la précédente.

L'ordre des options est indifférent.

AUTEURS

Franck LAURENT et Florian GOUDEY pour Christian GUILLEMINOT





Annexe 3 : Script xml2txt.pl

```
#!/usr/bin/perl
use strict; # always use strict
#use warnings;
use XML::Simple;
#use Data::Dumper;

my @resultat;  #tableau de message à rentrer dans le fichier
my $info=0;  #variable de trie des information à rentrer dans les résusltat
my @niveau;
my $index_niveau=0;  # ces 2 variables permettent de connaitre le niveau d'itération en cours
# gestion des paramètres
my $titre=0;  #affiche la ligne de titre
my $colonne=0;  #affiche la colonne de numérotation
my $attr=0;  #sortie compléte avec attribut
my $fichierXML;  #nom du fichier d'entrée
my $fichierTXT;  #nom du fichier de sorite
my $dossier;  #demande de traitement d'un dossier complet
my $type=0;  #défini si la ligne de titre doit être complétée par le niveau supérieur

#traitement des structures tableau
sub tableau{
  my $liste=$_[0];  #récupération de l'argument
  my $index=-1;  #index pour naviguer dans le tableau argument
  my $info_tmp=$info;
  foreach (@$liste)
  {
   #gestion du niveau d'itération
   $niveau[$index_niveau]++;
   if($index_niveau==1)
   {
    chop($resultat[1]);
    if($colonne==1)
    {
        $resultat[1].="\n$niveau[$index_niveau],";
    }
    else
    {
        $resultat[1].="\n";
    }
   }
   $index++;
   #traitement des données du tableau
   my $tmp1=$$liste[$index];
   if(ref($tmp1) eq "ARRAY")
   {
    $info=$info_tmp;
```






```
    $index_niveau++;
    tableau($tmp1);
    $index_niveau--;
   }
   if(ref($tmp1) eq "HASH")
   {
    $info=$info_tmp;
    $index_niveau++;
    hash_table($tmp1);
    $index_niveau--;
   }
  }
 $info=0;
}

#traitement des structures table de hachage
sub hash_table{
 my $ht=$_[0];
 foreach my $k (keys(%$ht))
 {
   #gestion du niveau d'itération
   $niveau[$index_niveau]++;
   if($index_niveau==1)
   {
    chop($resultat[1]);
   if($colonne==1)
    {
        $resultat[1].="\n$niveau[$index_niveau],";
    }
    else
    {
        $resultat[1].="\n";
    }
   }
   my $tmp1=$$ht{$k};
   #traitement des données de la table de hachage
   if(ref($tmp1) eq "ARRAY")
   {
    $info=1;
    $index_niveau++;
    tableau($tmp1);
    $index_niveau--;
   }
   if(ref($tmp1) eq "HASH")
   {
    $info=1;
    $index_niveau++;
    hash_table($tmp1);
```






```
    if($type==1)
    {
       $resultat[0].="$k,";
       $type=0;
    }
    $index_niveau--;
   }
   if(!ref($tmp1)&&$info==1)
   {
       if ($attr==0)
       {
         if($k eq "content")
         {
          $resultat[1].="$$ht{$k},";
          if($niveau[1]==1) #traitement des clés uniquement pour le premier élément pour
créer une ligne de titre
          {
             if(exists($$ht{'type'}))
             {
              $resultat[0].="$$ht{'type'},";
             }
             else
             {
              $type=1;
             }
          }
         }
       }
       else
       {
         $resultat[1].="$$ht{$k},";
         if($niveau[1]==1) #traitement des clés uniquement pour le premier élément pour
créer une ligne de titre
         {
          $resultat[0].="$k,";
         }
       }
   }
  }
  $info=0;
}

sub principal
{
  my $parser = XML::Simple->new(ForceContent => 1);  #création du parseur
  my $doc = $parser->XMLin($fichierXML);  #création de la structure de données issue du
fichier XML
  my $fichier=open(F1,">$fichierTXT")or die("open: $!");  #fichier de sortie
```






```
  hash_table($doc);  #traitement de la structure de données
  #traitement final de la ligne de titre
  if($colonne==1)
  {
   $resultat[0]="Numéro,$resultat[0]";
  }
  if($titre==1)
  {
   chop($resultat[0]);
   $resultat[0].="\n";
   print(F1 "$resultat[0]");
  }
  #traitement final des éléments à ressortir
  chop($resultat[1]);
  $resultat[1]=reverse($resultat[1]);
  chop($resultat[1]);
  $resultat[1]=reverse($resultat[1]);
  #intégration dans le fichier
  print(F1 "$resultat[1]");
  close(F1);
}

#initialisation pour fonctionnement en boucle
sub init
{
  @resultat=undef;
  @niveau=undef;
}

#générer les noms de fichier d'entrée et de sortie
sub nomFichier
{
  my $adresse=$_[0];
  $fichierXML=$adresse;
  my $i=0;
  my @char;
  my $char;
  for($char=chop($adresse);!($char eq "/")&&!($char eq "");$char=chop($adresse))
  {
     $char[$i]=$char;
     $i+=1;
  }
  $adresse.=$char;
  $fichierTXT=$adresse."resultat/";
  if(!(-e $fichierTXT))
  {
   open(tmp,"|mkdir ".$fichierTXT);
   close(tmp);
```

```
 }
 my $chemin=$fichierTXT;
 for(;$i>=0;$i-=1)
 {
   $fichierTXT.=$char[$i];
 }
 $fichierTXT.=".txt";
 return $chemin;
}

#traitement du tableau de paramètre
foreach my $a (@ARGV)
{
 if($a eq "t")  #recherche du paramètre concernant la ligne de titre
 {
  $titre=1;
 }
 elsif($a eq "c")  #recherche du paramètre concernant la colonne de numérotation
 {
  $colonne=1;
 }
 elsif($a eq "a")  #recherche du paramètre concernant le traitement des attributs XML
 {
  $attr=1;
 }
 elsif($a =~ m/\.xml$/ || $a =~ m/\.XML$/)  #recherche du paramètre concernant le fichier
d'entrée (extension en minuscule)
 {
  nomFichier($a);
 }
 elsif(-e $a)  #recherche du paramètre pour traiter un dossier complet
 {
  $dossier=$a;
  my $len=length($dossier);
  if (substr($dossier,$len-1,1) ne "/")
  {
   $dossier.="/";
  }
 }
 else   #Afficher une erreur si un argument et invalide
 {
  print "ERREUR2 : il faut rentrer le nom d'un fichier XML valide\n";
  exit;
 }
}
#forcer le mode traitement du dossier courant si rien n'est précisé
if(!(defined($dossier))&&!(defined($fichierTXT)))
{
```






```
  $dossier="./";
}

if(defined($dossier))
{
  my @liste=(<$dossier*.xml>,<$dossier*.XML>);
  foreach my $a (@liste)
  {
    nomFichier($a);
    principal();
    init();
  }
}
else
{
  principal();
}
close(tmp);

=head1 NOM

B<I<xml2txt.pl>> - Transforme les livres d'atomes XML de MPTK en fichiers text
exploitables par d'autres logiciels.

=head1 SYNOPSIS

B<xml2txt.pl> [t] [c] [a] [I<fichierXML>] [I<repertoire>]

=head1 DESCRIPTION

Sans argument cette commande traite l'intégralité fichiers XML du répertoire courant. Les
fichiers résultats sont générés dans un répertoire I<resultat> à l'adresse du ou des fichiers
XML traités. Ces fichiers portent le même nom que le fichier XML plus l'extension I<txt>

Les options suivantes sont possibles :

=over 5

=item t

Intègre une ligne de titre pour un traitement par tableur par exemple.

=item c

Intègre une colonne de numérotation pour un traitement par tableur par exemple.

=item a
```

Permet de traiter les attributs de balise XML en plus du contenu des balises.

=item I<fichierXML>

permet de ne traiter qu'un seul fichier. Il peut être indiqué en adresse relative ou absolue. La génération du résultat répond aux mêmes règles que dites précédemment. Cette option ne peut être utilisée qu'une seul fois.

=item I<repertoire>

permet de traiter un répertoire autre que le répertoire courant. La génération des résultats répond aux mêmes règles que dites précédemment. Cette option ne peut être utilisée qu'une seule fois et sans la précédente.

=back

L'ordre des options est indifférent.

=head1

AUTEURS

Franck LAURENT et Florian GOUDEY pour Christian GUILLEMINOT





Annexe 4 : Manuel du script MPTK1.praat

NOM

MPTK1.praat - Extrait les signaux sonores définis dans une TexGrid. Les signaux extraits sont directement traités par MPTK puis les livres obtenus, par xml2txt.pl.

SYNOPSIS

MPTK1.praat [destination] [commande MPTK] [commande xml2txt.pl]

DESCRIPTION

Le script supprime de la liste d'objets les signaux extraits pour éviter toute confusion en fin d'exécution. Le nom des signaux enregistrés sur le disque dur est composé d'un numéro (position dans la TextGrid), d'un underscore "_" et du commentaire donné dans la TextGrid. Les arguments permettent de rentrer les bonnes options aux logiciels tiers.

Les options suivantes sont possibles :

destination

Indiquez le dossier de destination des fichiers WAV créés. Si ce dossier n'existe pas il sera créé. Un dossier XML y est automatiquement créé pour ranger les résultats de MPTK. La commande xml2txt.pl crée le dossier résultat qui lui est propre dans le dossier XML.

commande MPTK

Indiquez la commande MPTK avec les options nécessaires à votre usage, à l'exception du livre d'atome et du fichier WAV à traité. Ces informations sont générées automatiquement.

commande xml2txt.pl

Indiquez la commande xml2txt.pl avec les options nécessaires à votre usage, à l'exception du dossier à traité.

AUTEURS

Franck LAURENT et Florian GOUDEY pour Christian GUILLEMINOT.





Annexe 5 : Scripts PRAAT

MPTK1.praat :

```
form Découpe du son
	comment destination
	text destination ~/projet/script_praat/test1/
	comment commande MPTK
	text commande_MPTK mpd -C /usr/local/bin/path.xml -D
	                                        ~/projet/script_praat/dico.xml -n 200
	comment commande xml2txt.pl
	text xml2txt ~/projet/script_praat/xml2txt.pl
endform
system mkdir -p 'destination$'/XML
echo __________
Extract non-empty intervals... 1 no
nb=numberOfSelected()
id = selected (1)
for i from 0 to 'nb'-1
	select 'id'+'i'
	fileName$=fixed$('i'+1,0)+"_"+selected$("Sound")
	file$ = destination$+"/"+fileName$+".wav"
	Write to WAV file... 'file$'
	Remove
	nomxml$ = destination$+"/XML/"+fileName$+".xml"
	mptk_suite$=commande_MPTK$+" "+file$+" - > "+nomxml$
	printline 'mptk_suite$'
	system_nocheck 'mptk_suite$'
	comxml$ = xml2txt$+" "+nomxml$
	printline 'comxml$'
	system_nocheck 'comxml$'
endfor
```





MPTK2.praat :

```
form Découpe du son
	comment destination
	text destination ~/projet/script_praat/test2/
	comment commande MPTK
	text commande_MPTK mpd -C /usr/local/bin/path.xml -D ~/projet/script_praat/dico.xml -n 200
	comment commande xml2txt.pl
	text xml2txt ~/projet/script_praat/xml2txt.pl
endform
system mkdir -p 'destination$'/XML
echo _________
Extract non-empty intervals... 1 no
nb=numberOfSelected()
id = selected (1)
for i from 0 to 'nb'-1
	select 'id'+'i'
	fileName$=fixed$('i'+1,0)+"_"+selected$("Sound")
	file$ = destination$+"/"+fileName$+".wav"
	Resample... 88200 1
	Write to WAV file... 'file$'
	plus 'id'+'i'
	Remove
	nomxml$ = destination$+"/XML/"+fileName$+".xml"
	mptk_suite$=commande_MPTK$+" "+file$+" - > "+nomxml$
	printline 'mptk_suite$'
	system_nocheck 'mptk_suite$'
	comxml$ = xml2txt$+" "+nomxml$
	printline 'comxml$'
	system_nocheck 'comxml$'
endfor
```





MPTK3.praat :

```
form Découpe du son
	comment destination
	text destination ~/projet/script_praat/test3/
	comment commande MPTK
	text commande_MPTK mpd -C /usr/local/bin/path.xml -D
	                              ~/projet/script_praat/dico.xml -n 200
	comment commande xml2txt.pl
	text xml2txt ~/projet/script_praat/xml2txt.pl
endform
system mkdir -p 'destination$'/XML
echo __________
Extract non-empty intervals... 1 no
nb=numberOfSelected()
id = selected (1)
for i from 0 to 'nb'-1
	select 'id'+'i'
	fileName$=selected$("Sound")
	file$ = destination$+"/"+fileName$+".wav"
	if fileReadable (file$)
		Append to existing sound file... 'file$'
	else
		Write to WAV file... 'file$'
	endif
	Remove
endfor
dest$=left$(destination$,1)
if dest$="~"
	home$ = environment$ ("HOME")
	destination$=replace$(destination$,dest$,home$,1)
endif
list$=destination$+"/*.wav"
Create Strings as file list... list 'list$'
nblist=Get number of strings
idlist = selected()
for i from 1 to 'nblist'
	fileName$=Get string... 'i'
	file$=destination$+"/"+fileName$
	nomxml$ = destination$+"/XML/"+fileName$+".xml"
	mptk_suite$=commande_MPTK$+" "+file$+" - > "+nomxml$
	printline 'mptk_suite$'
	system_nocheck 'mptk_suite$'
	comxml$ = xml2txt$+" "+nomxml$
	printline 'comxml$'
	system_nocheck 'comxml$'
endfor
Remove
```





Annexe 6 : Production de XML::SIMPLE

Exemple de fichier XML :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book nAtom="2" numChans="1" numSamples="262144"
sampleRate="44100" libVersion="0.5.6">
        <atom type="gabor">
                <par type="numChans">1</par>
                <support chan="0"><p>135168</p><l>16384</l></support>
                <par type="amp" chan="0">26.0189</par>
                <window type="gauss" opt="0"></window>
                <par type="freq">0.0854492</par>
                <par type="chirp">0</par>
                <par type="phase" chan="0">1.15969</par>
        </atom>
        <atom type="gabor">
                <par type="numChans">1</par>
                <support chan="0"><p>86784</p><l>16384</l></support>
                <par type="amp" chan="0">20.5322</par>
                <window type="gauss" opt="0"></window>
                <par type="freq">0.0477905</par>
                <par type="chirp">0</par>
                <par type="phase" chan="0">2.43945</par>
        </atom>
</book>
```





variable obtenue :

```
$VAR1 = {
          'sampleRate' => '44100',
          'numChans' => '1',
          'nAtom' => '2',
          'atom' => [
                      {
                        'support' => {
                                       'l' => {
                                                'content' => '16384'
                                              },
                                       'p' => {
                                                'content' => '135168'
                                              },
                                       'chan' => '0'
                                     },
                        'window' => {
                                      'opt' => '0',
                                      'type' => 'gauss'
                                    },
                        'par' => [
                                   {
                                     'content' => '1',
                                     'type' => 'numChans'
                                   },
                                   {
                                     'content' => '26.0189',
                                     'chan' => '0',
                                     'type' => 'amp'
                                   },
                                   {
                                     'content' => '0.0854492',
                                     'type' => 'freq'
                                   },
                                   {
                                     'content' => '0',
                                     'type' => 'chirp'
                                   },
                                   {
                                     'content' => '1.15969',
                                     'chan' => '0',
                                     'type' => 'phase'
                                   }
                                 ],
                        'type' => 'gabor'
                      },
                      {
                        'support' => {
                                       'l' => {
                                                'content' => '16384'
                                              },
                                       'p' => {
                                                'content' => '86784'
                                              },
                                       'chan' => '0'
                                     },
                        'window' => {
```






```
                    'opt' => '0',
                    'type' => 'gauss'
                  },
         'par' => [
                  {
                    'content' => '1',
                    'type' => 'numChans'
                  },
                  {
                    'content' => '20.5322',
                    'chan' => '0',
                    'type' => 'amp'
                  },
                  {
                    'content' => '0.0477905',
                    'type' => 'freq'
                  },
                  {
                    'content' => '0',
                    'type' => 'chirp'
                  },
                  {
                    'content' => '2.43945',
                    'chan' => '0',
                    'type' => 'phase'
                  }
                ],
         'type' => 'gabor'
       }
     ],
  'libVersion' => '0.5.6',
  'numSamples' => '262144'
};
```





Annexe 7 : Exemple de TextGrid

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 298.48185941043084
tiers? <exists>
size = 1
item []:
    item [1]:
        class = "IntervalTier"
        name = "son"
        xmin = 0
        xmax = 298.48185941043084
        intervals: size = 145
        intervals [1]:
            xmin = 0
            xmax = 30.59143556155563
            text = ""
        [...]
        intervals [26]:
            xmin = 74.10572390974833
            xmax = 74.46506006045982
            text = "ad\9ea"
        intervals [27]:
            xmin = 74.46506006045982
            xmax = 77.39795330405028
            text = ""
        intervals [28]:
            xmin = 77.39795330405028
            xmax = 77.71752826354505
            text = "ad\9ea"
        intervals [29]:
            xmin = 77.71752826354505
            xmax = 80.75494669810018
            text = ""
        intervals [30]:
            xmin = 80.75494669810018
            xmax = 81.07337687003871
            text = "ad\9ea"
       [...]
        intervals [145]:
            xmin = 291.3001956699943
            xmax = 298.48185941043084
            text = ""
```